

INITIAL PROCESSING LOAD SCATTERING SYSTEM FOR MULTI-CPU CONSTITUTION SYSTEM

Patent number: JP10011412
Publication date: 1998-01-16
Inventor: AMAGASAKI KENJI
Applicant: NIPPON DENKI IDO TSUSHIN KK
Classification:
- international: G06F15/177; G06F15/16
- european:
Application number: JP19960181663 19960624
Priority number(s): JP19960181663 19960624

[View INPADOC patent family](#)

Abstract of JP10011412

PROBLEM TO BE SOLVED: To quickly start the scattering system without impairing the starting order by dividing the initial processing into blocks of every function that can be processed in parallel to each other and assigning these processing of divided blocks to all CPUs. **SOLUTION:** The initial processing is divided into blocks of every function that can be processed in parallel to each other. For instance, the check/clear processing of local memories 2-1 to 2-5 used for initial processing in common to all CPUs and the processing which loads the programs in the memories 2-1 to 2-5 from the system data are set in a block A. Then the processing is set in a block B to transfer the information contents of a common memory 3-2 under waiting to a backup memory 6. When the system is reset, the CPU 2-1 to 2-5 are also reset. Then the block A of the processing common to all CPUs is carried out. When the processing of the block A is completed, a CP# setting processing is carried out based on a CP# management table that is set in the memory 6.

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平10-11412

(43)公開日 平成10年(1998) 1月16日

(51)Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 15/177			G 0 6 F 15/16	4 2 0 S
15/16	3 8 0			3 8 0 Z

審査請求 有 請求項の数3 F D (全 9 頁)

(21)出願番号 特願平8-181663

(22)出願日 平成8年(1996) 6月24日

(71)出願人 390000974

日本電気移動通信株式会社
横浜市港北区新横浜三丁目16番8号 (N
E C移動通信ビル)

(72)発明者 尼崎 健治

神奈川県横浜市港北区新横浜三丁目16番8
号 日本電気移動通信株式会社内

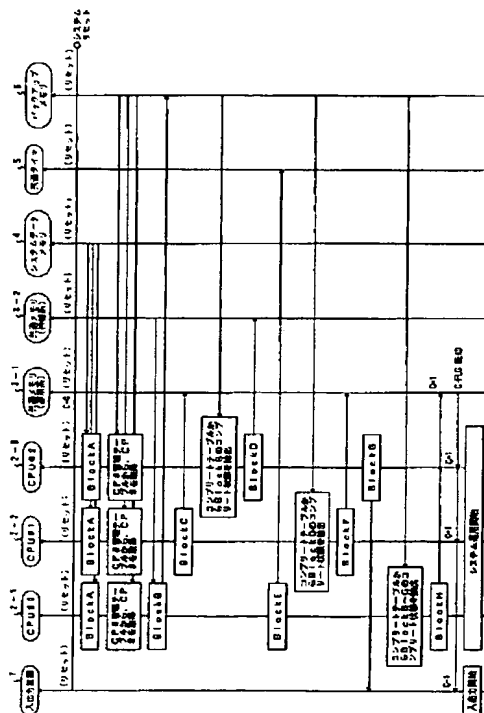
(74)代理人 弁理士 高橋 友二

(54)【発明の名称】 マルチCPU構成システムにおける初期処理負荷分散方式

(57)【要約】

【課題】 システムの立ち上げ順序をそこなうことなく、迅速なシステムの立ち上げを可能とするマルチCPU構成システムにおける初期処理負荷分散方式を提供する。

【解決手段】 マルチCPUにて構成されるシステムの起動時に行われる初期処理負荷を分散する方式であって、初期処理を平行に処理できる機能ごとにいくつかのブロックに分割し、実装されているすべてのCPU 2-1~2-3にブロック化した処理を割り当てることにより、初期処理の負荷を分散する。



【特許請求の範囲】

【請求項1】 マルチCPUにて構成されるシステムの起動時に行われる初期処理負荷を分散する方式であって、初期処理をパラレルに処理できる機能ごとにいくつかのブロックに分割し、実装されているすべてのCPUにブロック化した処理を割り当てることにより、初期処理の負荷を分散することを特徴とするマルチCPU構成システムにおける初期処理負荷分散方式。

【請求項2】 前記CPUに対するブロック化した処理を割り当てるに際し、コンプリート状態を確認することで、システムの立ち上げの順序をそこなわないようにした請求項第1項記載のマルチCPU構成システムにおける初期処理負荷分散方式。

【請求項3】 前記ブロック化した初期処理が、各CPU共通初期処理のローカルメモリのチェック及びクリア処理、システムデータからプログラムをローカルメモリへロードする処理、共通メモリ（待機系）の情報内容をバックアップメモリへ転送する処理、共通メモリ（運用系）のチェック及びクリア処理、共通メモリ（待機系）のチェック及びクリア処理、共通タイマのタイマクリア処理、共通データテーブルの初期化処理、入出力装置の初期化処理、初期処理完了フラグのON処理のいずれかである請求項第1項または第2項記載のマルチCPU構成システムにおける初期処理負荷分散方式。

【発明の詳細な説明】**【0001】**

【発明の属する技術分野】 本発明は、マルチCPU構成システムにおける起動時間の短縮化を図るための初期処理負荷分散方式に関するものである。

【0002】

【従来の技術】 従来から、処理時間の短縮を図るために、処理を分割したグループCPUで並列に行なうマルチCPU構成システムは、例えば特開平5-120245号公報に開示されているように良く知られている。図1はそのような従来のマルチCPUシステムの構成の一例を示す図である。なお、本発明の対象となるマルチCPU構成システムは、図1に示す従来のものと同様である。

【0003】 図1に示す例において、マルチCPU構成システム1は、所望の実装枚数、ここでは5枚のCPU2-1～2-5と、動的に変化するデータを格納しておく運用系および待機系の2系列からなる共通メモリ3-1、3-2と、プログラム、システムデータ等の静的なデータを格納しておくシステムデータメモリ4と、シーケンス管理上必要なタイマを管理する共通タイマ5と、共通メモリ3-1または3-2のデータをバックアップしておくためのバックアップメモリ6と、外部とのインターフェースを担う入出力装置7とから構成されている。なお、8-1～8-2は各CPU2-1～2-5に設けられたローカルメモリである。

【0004】 図2は図1に示すマルチCPU構成システム1の起動時に行われる初期処理の状態を説明するための図である。図2において、まずシステムリセットが行われると、システムを構成する各装置がリセットされる。各々のCPU2-1～2-5は、各CPUが具備するローカルメモリ8-1～8-5のチェック・クリアを行い、システムデータメモリ4にあるプログラムのロードを行う。次に、運用系の共通メモリ3-1上にあるinitial-FLAG（初期処理フラグ、I-FLAGと記す）の値を読み出し、このI-FLAGの値が「0」であったならば、そのCPU、ここではCPU2-1がシステムの初期処理を担当するマスタCPUとなる。また、このI-FLAGの値が「1」であったならば、その他のCPU（ここではCPU2-5を例示する）は、マスタCPUがシステムの初期処理を完了するまで待つスレーブCPUとなる。

【0005】 マスタCPUとなったCPU2-1は、順に、待機系の共通メモリ3-2のデータをバックアップメモリ6へ待避させる待避処理を行い、運用系共通メモリ3-1と待機系共通メモリ3-2のチェック・クリア処理を行ない、共通タイマ5のクリア処理を行ない、共通メモリ3-1上のデータテーブルの初期化を行ない、入出力装置7の初期化を行ない、共通メモリ3-1上にあるcomplete-FLAG（初期処理完了フラグ、C-FLAGと記す）に「1」を設定する。そして、C-FLAGが「1」になったことをスレーブCPU2-5と入出力装置7が認識後、システムが運用開始となる。

【0006】

【発明が解決しようとする課題】 上述した従来のマルチCPU構成システム1では、初期処理を1枚のCPU（マスタCPU）2-1のみで行っているため、プログラムのバージョンアップ、システムデータの変更等に伴うシステムリセットが行われた際、システムが再度運用を開始するまでの時間がかかりかかるという問題があった。また、システムが運転再開するまでの時間、ユーザに対してサービスを提供することができないため、顧客満足度が低下する問題もある。さらに、今後の良質なサービスの提供のためにプログラムおよび共通メモリの増大が見込まれており、一層初期処理時間が増大する可能性が高く、そのような場合上述した問題がより一層顕著になる。

【0007】 本発明の目的は上述した課題を解消して、システムの立ち上げ順序をそこなうことなく、迅速なシステムの立ち上げを可能とするマルチCPU構成システムにおける初期処理負荷分散方式を提供しようとするものである。

【0008】

【課題を解決するための手段】 本発明のマルチCPU構成システムにおける初期処理負荷分散方式は、マルチCPUにて構成されるシステムの起動時に行われる初期処

理負荷を分散する方式であって、初期処理をパラレルに処理できる機能ごとにいくつかのブロックに分割し、実装されているすべてのCPUにブロック化した処理を割り当てることにより、初期処理の負荷を分散することを特徴とするものである。

【0009】具体的に本発明のマルチCPU構成システムにおける初期処理負荷分散方式は、マルチCPUにて構成されるシステムの起動時に行われる初期処理負荷を分散する方式であって、初期処理をパラレルに処理できる機能ごとにいくつかのブロックに分割し、実装されているすべてのCPUにブロック化した処理を割り当てることにより、初期処理の負荷を分散することを特徴とする。

【0010】また、前記CPUに対するブロック化した処理を割り当てるに際し、コンプリート状態を確認することで、システムの立ち上げの順序をそこなわないようにしたことを特徴とする。

【0011】さらに、前記ブロック化した初期処理が、各CPU共通初期処理のローカルメモリのチェック及びクリア処理、システムデータからプログラムをローカルメモリへロードする処理、共通メモリ（待機系）の情報内容をバックアップメモリへ転送する処理、共通メモリ（運用系）のチェック及びクリア処理、共通メモリ（待機系）のチェック及びクリア処理、共通タイマのタイマクリア処理、共通データテーブルの初期化処理、入出力装置の初期化処理、初期処理完了フラグのON処理のいずれかであることを特徴とする。

【0012】

【発明の実施の形態】以下、本発明の一実施例を図面を参照して説明する。図3は本発明のマルチCPU構成システムにおける初期処理負荷分散方式における初期処理のブロック化の状態を示す図である。図3では、図2に示す従来のマルチCPU構成システムの初期処理を分割して、処理できる機能ごとにブロック化を図っている。図3に示す例において、ブロックAには、各CPU共通初期処理のローカルメモリ8-1～8-5のチェック及びクリア処理およびシステムデータからプログラムをローカルメモリ8-1～8-5へロードする処理を設定し、ブロックBには、待機系の共通メモリ3-2の情報内容をバックアップメモリ6へ転送する処理を設定し、ブロックCには、運用系の共通メモリ3-1のチェック及びクリア処理を設定し、ブロックDには、待機系の共通メモリ3-2のチェック及びクリア処理を設定し、ブロックEには、共通タイマ5のタイマクリア処理を設定し、ブロックFには、運用系の共通メモリ3-1内の共通データテーブルの初期化処理を設定し、ブロックGには、入出力装置7の初期化処理を設定し、ブロックHには、初期処理完了フラグONを設定する。

【0013】図4は本発明の初期処理分散方式を用いたマルチCPU構成システムの起動時に行われる初期処理

の状態の一例を示す図である。図4に示す例では、CPU2-1、2-2、2-3の3枚を実装した例を示している。まず、図4に示すマルチCPU構成システムにおいて、システムリセットが実行されると、CPU2-1～2-3に対しリセットがかかる。その後、図3に示す各CPU共通処理のブロックAを実行する。ブロックAの処理が終了した後、バックアップメモリ6上に設定されている図7に示すCP#管理テーブルを利用して、図8に示すCP#設定処理を実行する。

【0014】図8に従ってCP#設定処理を説明すると、まず図7に示すCP#管理テーブルからLOCKを取得する（801）。LOCKの内容が「OFF」であれば、次の処理を行ない、「ON」であれば再度LOCKの取得を行なう（802）。次に、LOCKに「ON」を設定する（803）。そして、図7に示すCP#管理テーブルからCP#を取得し（804）、CP#が5未満であるかの判断を行ない（805）、5未満であれば取得したCP#を1インクリメントし、テーブルに設定する（806）。5以上であれば次の処理を行なう。再度にLOCKを「OFF」に設定する（807）。ここでは、CPU2-1がCP#0に、CPU2-2がCP#1に、CPU2-3がCP#2に設定されている。

【0015】次に、各CPUは取得したCP#に基づき、図5に示すブロック処理ディスパッチ（割当）一覧表によって、処理すべきブロックを判定する。すなわち、バックアップメモリ6上に設定されている図6に示すコンプリートテーブルを利用して、図9に示すコンプリート状態の設定処理を実行する。

【0016】図9に従ってコンプリート状態設定処理を説明すると、CP#0のCPU2-1は、図5に示す一覧表におけるCPU実装数3の欄のCP#0の部分に基づいて、ブロックBの図6に示すコンプリートテーブルのLOCKを取得する（901）。LOCKの内容が「OFF」であれば次の処理を行ない、「ON」であれば再度LOCKの取得を行なう（902）。LOCKに「ON」を設定する（903）。そして、ブロックBの図6に示すコンプリートテーブルのコンプリート状態を取得し（904）、コンプリート状態が「未実施」であれば（905）、コンプリート状態に「実施中」を設定する（906）。コンプリート状態が「未実施」以外の場合は、異常処理を行なう。次に、CP#に現在のCP#ここでは「0」を設定する（909）。最後にLOCKに「OFF」を設定する（910）。その後、CP#0のCPU2-1においてブロックBの処理を行ない、処理終了後、図9のコンプリート状態設定処理に従って、コンプリート状態「実施中」（907）をコンプリート状態「完了」に設定する（908）。

【0017】上記と同様に、CP#1のCPU2-2はブロックCの処理を行なう。CP#2のCPU2-3

は、図5に示すブロック処理ディスパッチ一覧表に従ってブロックDの処理を行なう様に判断できるが、コンプリート条件があるのでブロックBのコンプリート状態が「完了」になった後に処理を開始する。

【0018】ここで各CPUは、ブロックAの後の一回目の処理が終了したので、再度、図7に示すCP#管理テーブルからCP#を取得し、図5に示すブロック処理ディスパッチ一覧表のCPU実装数3の部分判定し、CP#0のCPU2-1はブロックEの処理を実行し、CP#1のCPU2-2はブロックDのコンプリート状態が「完了」になった後ブロックFの処理を実行し、CP#2のCPU2-3はブロックGの処理を実行する。

【0019】最後に、CP#0のCPU2-1がブロックB～Gのコンプリート状態を判定し、すべて「完了」であれば、ブロックHの処理を実行する。図5に示す各CP#の処理終了後のCPU2-1～2-3とブロックG終了後の入力装置7は、共通メモ3-1上にあるC-FLGが「1」になるまで取得を行って、「1」になったことを認識後、システムが運用開始となる。

【0020】

【発明の効果】以上の説明から明らかなように、本発明のマルチCPU構成システムにおける初期処理負荷分散方式によれば、初期処理がブロック化され複数のCPUによってパラレルに処理実行されることにより、システムの立ち上げ順序をそこなうことなく、迅速なシステムの立ち上げが可能となる。

【図面の簡単な説明】

【図1】本発明の対象となるマルチCPU構成システムの一例の構成を示すブロック図である。

【図2】マルチCPU構成システムにおける従来の初期処理の状態を示す図である。

【図3】マルチCPU構成システムにおける初期処理のブロック化の一例を示す図である。

【図4】本発明の初期処理負荷分散方式を用いたマルチCPU構成システムにおける初期処理の状態を示す図である。

【図5】ブロック処理のディスパッチ一覧表の一例を示す図である。

【図6】コンプリートテーブルの一例を示す図である。

【図7】CP#管理テーブルの一例を示す図である。

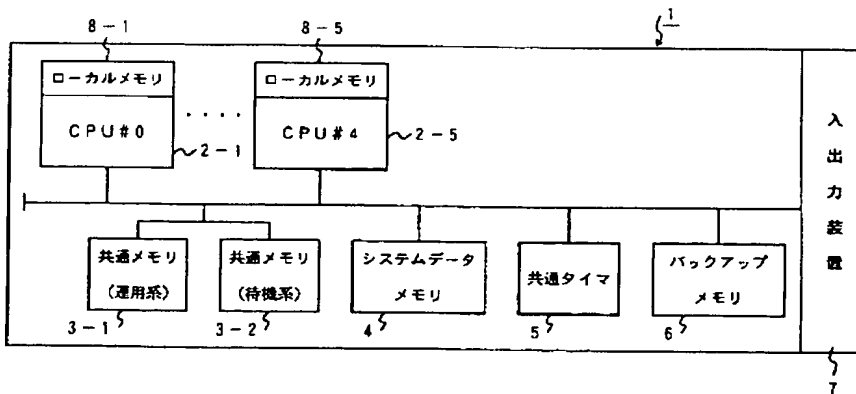
【図8】CP#設定処理の一例を示すフローチャートである。

【図9】コンプリート状態設定処理の一例を示すフローチャートである。

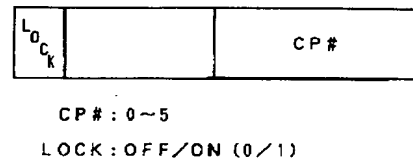
【符号の説明】

- 1 マルチCPU構成システム
- 2-1～2-5 CPU
- 3-1, 3-2 共通メモリ
- 4 システムデータメモリ
- 5 共通タイマ
- 6 バックアップメモリ
- 7 入出力装置

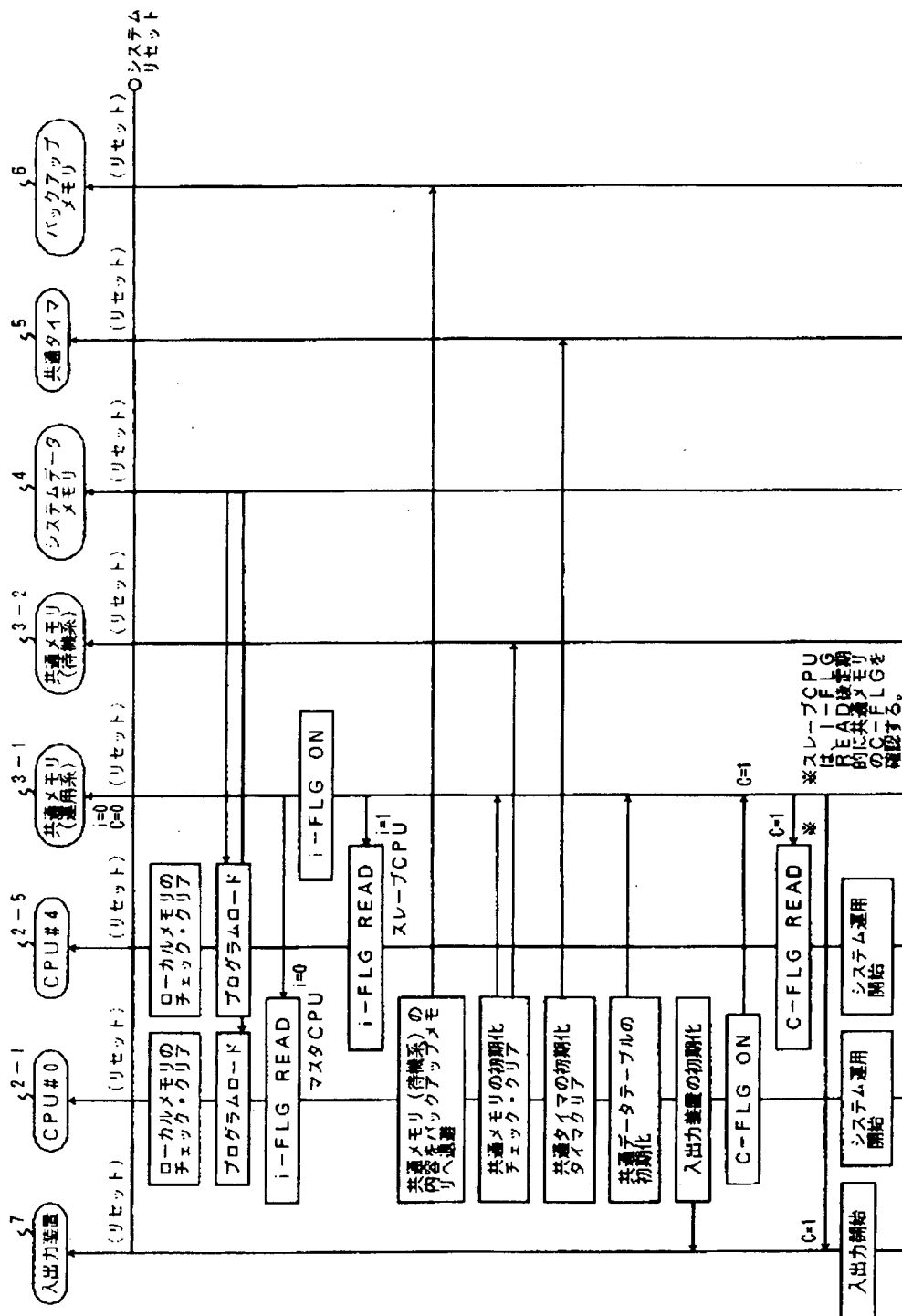
【図1】



【図7】



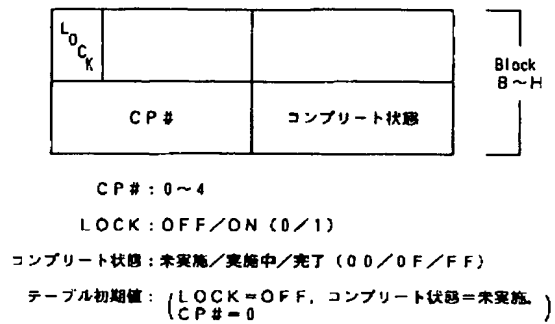
【図2】



【図3】

ブロックA	<ul style="list-style-type: none"> ローカルメモリのチェック及びクリア システムデータからプログラムをローカルメモリへロードする
ブロックB	<ul style="list-style-type: none"> 共通メモリ（待機系）の情報内容をバックアップメモリへ転送
ブロックC	<ul style="list-style-type: none"> 共通メモリ（運用系）のチェック及びクリア
ブロックD	<ul style="list-style-type: none"> 共通メモリ（待機系）のチェック及びクリア
ブロックE	<ul style="list-style-type: none"> 共通タイマのタイマクリア
ブロックF	<ul style="list-style-type: none"> 共通データテーブルの初期化
ブロックG	<ul style="list-style-type: none"> 入出力装置の初期化
ブロックH	<ul style="list-style-type: none"> 初期処理完了フラグON

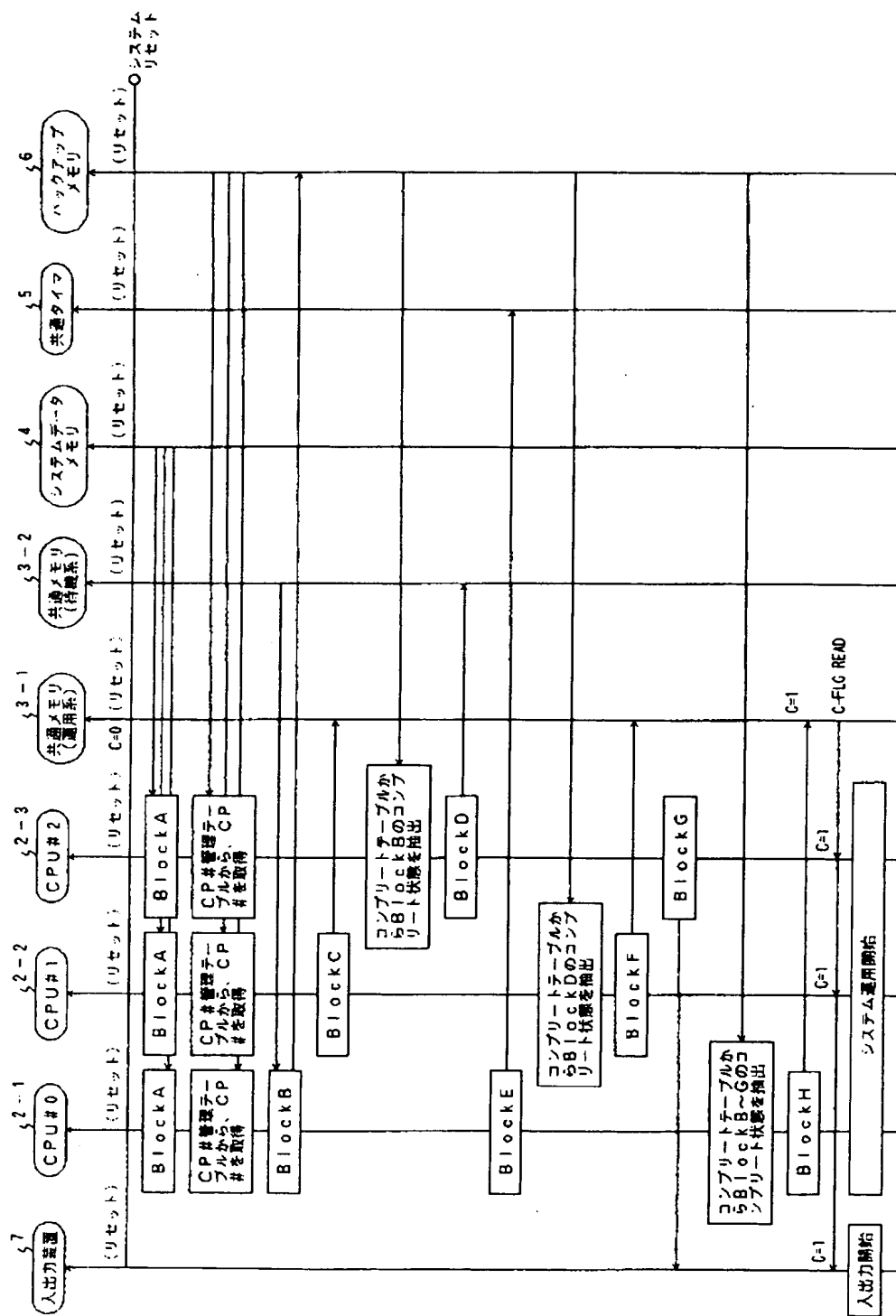
【図6】



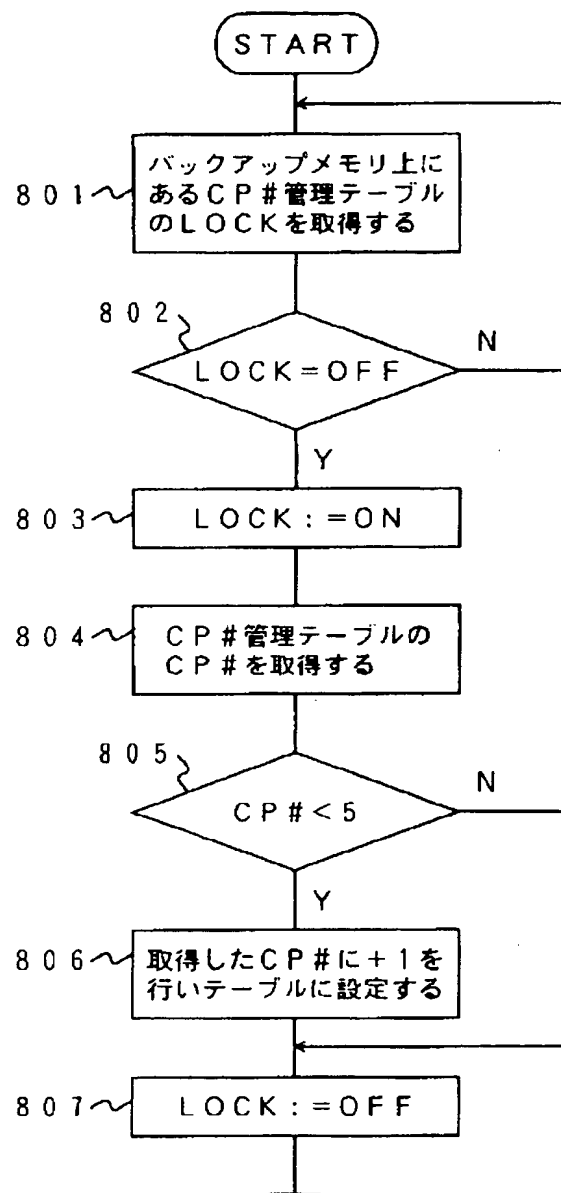
【図5】

Block # CP #	B	C	D	E	F	G	H	CPU実数値
CP0	○	○	○	○	○	○	○	1
CP0	○		○		○		○	2
CP1		○		○		○		
CP0	○			○			○	3
CP1		○			○			
CP2			○			○		
CP0	○				○			4
CP1		○				○		
CP2			○				○	
CP3				○				
CP0	○					○		5
CP1		○					○	
CP2			○					
CP3				○				
CP4					○			
コンプリート条件	なし	なし	BlockB コンプリート	なし	BlockD コンプリート	なし	BlockB~G コンプリート	

【図4】



【図8】



【図9】

